HoGent

BEDRIJF
EN
ORGANISATIE

Hoofdstuk 21:
GEGEVENSSTRUCTUREN

INLEIDING

- **Reeds gezien :**
  gegevensstructuren met een vaste lengte

  - Eéndimensionele arrays
  - Meerdimensionele arrays

INLEIDING

- **H21: Dynamische gegevensstructuren**
  - Variabele lengte: lengte kan kleiner en groter worden tijdens runtime.
  - Enkele dynamische gegevensstructuren:
    - Gelinkte lijsten
    - Stacks
    - Queues
    - Binaire bomen

ZELF-REFERENTIE KLASSE

- Zelf-referentie klasse
  - Bevat een instantie-variabele die refereert naar een object van dezelfde klasse.

Bv:
```
class Node { // Node = knoop
    private int data;
    private Node next;
    public Node (int data) {…} // constructor
    public final void setData (int data) {…}
    public int getData() {…}
    public final void setNext (Node next) {…}
    public Node getNext() {…}   }
```

ZELF-REFERENTIE KLASSE

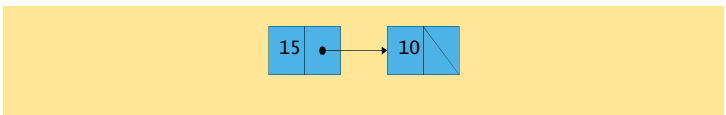
- **private Node next;**

  De instantie-variabele nextNode wordt een *"link"* genoemd.
  - nextNode verbindt *("links")* een Node object met een ander Node object.

ZELF-REFERENTIE KLASSE

Programma's kunnen "zelf-referentie objecten" verbinden. Zo kan er een gegevensstructuur gecreëerd worden zoals lijsten, queues, stacks en bomen.

Bv:



DYNAMISCHE GEHEUGENTOEWIJZING

- **Dynamische geheugentoewijzing**
  - Als we objecten aan de dynamische gegevensstructuur toevoegen, verzoeken we het systeem om meer geheugen.
  - Als we een object niet meer nodig hebben, geven we de ruimte aan het systeem terug:
    - JAVA heeft een automatische garbage collection.